الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد (م.و.ت.م.ب)

COURS DE LANGAGE PASCAL

SÉRIE 04

LES AUTRES TYPES DÉFINIS PAR L'UTILISATEUR

OBJECTIF PÉDAGOGIQUE : À la fin de cette série, le stagiaire doit être capable d'utiliser les règles d'écriture du programme pascal.

PLAN DE LA LEÇON :

INTRODUCTION :

Les types prédéfinis permettent de manipuler aisément des nombres ou des caractères. Mais, vous pouvez être amenés à traiter d'autres sortes d'informations, par exemple :

- Des mois de l'année (JANVIER, FEVRIER, ...)
- Des jours de semaines (SAMEDI, DIMANCHE, ...)
- Des formules chimiques
- Des marques de voitures
- Des indications d'état civil (CELIBATAIRE, MARIE, DIVORCE, ...)

En Pascal, comme dans d'autres langages, il est possible de coder les informations ; Ainsi vous pouvez convenir que 1 représente JANVIER, 2 représente FEVRIER, ...etc. Pascal offre la possibilité de manipuler ces informations d'une manière plus naturelle en leur attribuant le nom de votre choix ; Par exemple JANVIER, FEVRIER, ... pour des noms de mois, ou simplement JAN, FEV, ... si vous le souhaitez.

Pour ce faire, il vous suffira de définir ce que l'on appelle « un type défini par ENUMERE » (ou plus brièvement « type énuméré ») c'est à dire un type dans lequel vous énumérez chacune des valeurs possibles.

D'autre part, il arrive que l'on ait à manipuler des informations dont les valeurs ne peuvent couvrir qu'un intervalle limité de valeurs.

EXEMPLE :

Un âge pourrait être un entier entre 0 et 100.
Une note d'élève pourrait être un réel compris entre 0 et 20.
Une lettre alphabétique pourrait être un caractère compris entre 'A' et 'Z'.

I- LE TYPE ÉNUMÈRE :

1- Introduction au type énumère :

Considérons le cas où nous souhaitons manipuler des jours de semaine en les désignant simplement par leur nom. Nous commencerons par définir le nouveau type, comme suit :

TYPE Jour = (SAMEDI, DIMANCHE, LUNDI, MARDI, MERCREDI, JEUDI, VENDREDI) ;

Cette instruction définit un nouveau type désigné par l'identificateur ' Jour', elle énumère les différentes valeurs permises pour le type, chacune de ces valeurs est un identificateur de votre choix.

Pour l'instant, nous ne disposons pas encore d'une variable du type 'Jour' ; Pour ce faire il nous suffit d'en déclarer comme à l'accoutumé. Ainsi l'instruction :

VAR début, fin, date : Jour ; crée trois variables nommées début, fin et date de type Jour.

Que pourrons-nous faire avec des variables d'un tel type. Tout d'abord, nous pourrons leur assigner des valeurs, par affectation telle que :

Début = MARDI ;
Date = début ;

La première affecte à la variable 'début' la valeur MARDI, la seconde affectation quant à elle, assigne à la variable 'date' la valeur de la variable 'début'.

Nous pourrons également comparer des valeurs du type Jour :

IF date = DIMANCHE THEN....
IF date > MERCREDI THEN....

La première comparaison est vrai, si la valeur de la variable ‘date’ est : DIMANCHE. Par contre, la seconde utilise l’expression booléenne ‘ date > MERCREDI ’ qui sera vrai si la valeur de ‘date’ est postérieur à MERCREDI en utilisant l’ordre dans lequel les valeurs ont été déclarées dans la définition du type.

Exemple :

```
PROGRAM ENUM ;

TYPE
Jour = {SAMEDI, DIMANCHE, LUNDI, MARDI, MERCREDI, JEUDI, VENDREDI} ;
VAR     date: jour;
BEGIN
FOR date: = SAMEDI TO VENDREDI
      DO
        BEGIN
        WRITELEN (‘Voici un nouveau jour’, date) ;
        IF date = LUNDI
           THEN    WRITELEN (‘Aujourd’hui, les enseignants sont en congé’) ;
        IF date =MERCREDI
            THEN      WRITELEN (‘C’est le dernier jour de travail’) ;
        IF date = JEUDI    OR    date = VENDREDI
            THEN   WRITELEN (‘on se repose’);

     END;
END.
```

Exécution de l'algorithme

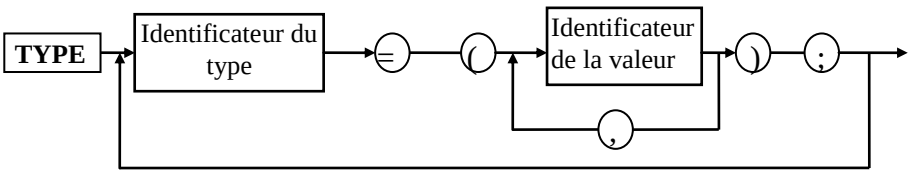
```
Voici un nouveau jour SAMEDI
Voici un nouveau jour DIMANCHE
Voici un nouveau jour LUNDI
Aujourd'hui, les enseignants sont en congé
Voici un nouveau jour MARDI
Voici un nouveau jour MERCREDI
C'est le dernier jour de travail
Voici un nouveau jour JEUDI
On se repose
Voici un nouveau jour VENDREDI
On se repose
```

Vous constatez que nous avons utilisé la variable 'date' (du type jour) comme variable de contrôle d'une instruction FOR. Cette dernière répète le bloc d'instructions quelle renferme en donnant successivement à la variable 'date', les valeurs SAMEDI, DIMANCHE, ..., JEUDI, VENDREDI ; l'ordre de parcours de ces valeurs étant celui fourni par la déclaration du type 'Jour'.

2- Syntaxe et règles d'écritures :

Une instruction TYPE peut comporter une ou plusieurs définitions de types (séparés alors par des points-virgules). Nous verrons plus tard d'autres sortes de types que les types définis par ENUMERE.

Le diagramme de syntaxe est :



REMARQUE :

- N'utilisez pas des mots réserves comme identificateur.

Exemple : Vous ne pouvez pas écrire :

TYPE Note = {DO, RE, MI, FA, SOL, LA, SI} car DO est un mot réservé.

- Un même identificateur ne peut pas désigner plusieurs choses différentes. Il n'est pas possible d'utiliser des identificateurs ayant le même nom dans deux types différents, par exemple **Samedi** et **Mercredi** dans les déclarations suivantes :

TYPE Semaine = (Samedi, Dimanche, Lundi, Mardi, Mercredi) ;
Jour = (Mercredi, Jeudi, Vendredi, Samedi) ;

En effet, toute donnée décrite dans un type ENUMERE est ordonnée et prend donc une valeur (calculée par la fonction ORD). Ainsi, dans la première déclaration Samedi < Dimanche < Lundi < Mardi < Mercredi ou bien ORD (Samedi) = 0, ORD (Dimanche) = 1, ORD(Lundi) = 2, ORD(Mardi) = 3, ORD(Mercredi) = 4. Par contre dans la seconde, ORD(Mercredi) = 0,
ORD(Jeudi) = 1, ORD(Vendredi) = 2, ORD(Samedi) = 3.

- Un identificateur n'est pas une constante, autrement dit, il n'est pas question d'en déclarer un type énuméré de nombres comme : TYPE impair = {1, 3, 5, 7, 9, 11} ; ou de caractères comme : TYPE voyelle = { 'a', 'e', 'o', 'i' , 'u' , 'y'} ;

3- Propriétés des types énumèrent :

Comme nous l'avons mentionné, les types ENUMERES sont des types ordinaux à part entière. Les principales conséquences en sont les suivantes :

- Ils sont ordonnés, l'ordre est défini comme étant celui dans lequel vous avez ENUMERE les différentes valeurs du type.

- Les opérateurs relationnels : =, <, >, >=, <=, <> sont utilisés pour comparer les éléments du même type.

- Les fonctions ORD(x), PRED(x) et SUCC(x) sont utilisables et qui représentent respectivement l'ordre, le prédécesseur et le successeur d'une valeur x du type. Ainsi, avec notre déclaration du type 'jour', SUCC(Mardi) vaut Mercredi et PRED (jeudi) vaut mercredi. PRED(Samedi) et SUCC(Vendredi) ne sont pas définis, quant à la fonction ORD il faut noter que la première valeur du type possède le rang 0, la deuxième le rang 1, ...etc. Ainsi, dans notre type 'jour', ORD(Samedi) vaut 0 tandis que ORD(Vendredi) vaut 6.

- Les valeurs du type sont énumérables. Autrement dit des valeurs d'un type ENUMERE peuvent apparaître comme variables de contrôle d'une instruction FOR ou dans une instruction CASE. Ainsi, avec notre type 'jour' comme nous l'avons défini : jour (3) a la valeur LUNDI et jour (5) a la valeur MERCREDI.

Il vous parait certainement naturel de disposer d'instructions comme READ et WRITE pour échanger de l'information avec votre programme. Néanmoins, celles-ci ne fonctionnent qu'avec certains types prédéfinis.

Si vous cherchez à afficher une variable d'un tel type en écrivant par exemple : WRITE (date) où date a été déclarée de type jour, vous obtiendrez à la compilation le message : **error 66 I/O are**

not allowed qui signifie : Les entrées-sorties ne sont pas autorisées (pour ce type)

Pour y remédier, il reste bien sûr possible d'échanger des informations d'un tel type en prévoyant des instructions assurant en quelque sorte leur codage.

Exemple :

Voici une instruction CASE permettant d'afficher en clair, le contenu d'une variable nommée date, supposée être du type 'jour' défini précédemment :

```
CASE   date   OF

        SAMEDI    : Write ('Samedi');
        Dimanche  : Write ('Dimanche');
        Lundi     : Write ('lundi');
        Mardi     : Write ('Mardi');
        MERCREDI : Write ( 'Mercredi') ;
        JEUDI     : Write ( 'Jeudi') ;
        VENDREDI : Write ( 'Vendredi') ;

ENDCASE ;
```

- Les opérations de vérification des affectations à priori, une variable de type ENUMERE ne doit pouvoir se voir affecter que des valeurs spécifiées dans la définition du type. Il est vrai que toute tentative d'affectation de la valeur d'une expression d'un type différent sera systématiquement rejetée lors de la compilation.

Cependant, commencez par examiner ce programme et les résultats qu'il fournit :

PROGRAM Essai ; TYPE Jour = (Samedi, Dimanche, Lundi, Mardi, Mercredi, Jeudi, Vendredi) ; VAR Date, Demain : Jour ; BEGIN Date := Vendredi ; Demain := SUCC (Date) ; WRITELN (‘Ordre de demain :’ , ORD (Demain) ; END.	**Exécution** Ordre de demain:7

Pour le compilateur, une expression telle que SUCC (date) est du type de la variable date ; Il ne décèle donc aucune erreur.

Pour l’exécution, la fonction SUCC, se contente de passer d’un élément au suivant en ajoutant ‘1’ à la valeur de la fonction ORD. Pour la valeur Vendredi, il n’existe pas de successeur et l’exécution du programme est interrompue par l’erreur d’une tentative d’écriture d’une valeur inconnue.

II-LE TYPE INTERVALLE :

1- Introduction au type intervalle :

Abordons maintenant la deuxième sorte de type défini par l’utilisateur à savoir le type intervalle.

Le type intervalle a pour principe de fixer le champ de variation d’une variable de n’importe quelle type scalaire, défini par l’utilisateur ou prédéfini, à l’exception du type réel (car le nombre de valeurs différentes est illimité) en en précisant les bornes : inférieure et supérieure.

Exemple :

Déclarer un type nommé âge dont les valeurs seraient des entiers compris entre 0 et 120.

TYPE âge = 1... 120 ;
ou encore

CONST âge-max = 120 ;
TYPE âge = 1... âge _ max ;

Il est alors possible ensuite de déclarer des variables de type âge.

VAR âge père, âge mère, courant : âge ;

Contrairement au type ENUMERE, les valeurs du type intervalle sont simplement une plage de valeurs d'un type ordinal déjà défini (la notion d'ordre est importante, car les éléments de l'intervalle sont ordonnés), Ici le type Entier. Donc les variables du type âge pourront être manipulées de la même manière que des variables entières.

Exemple :

READ (âge père, âge mère) ;
n := âge père + âge mère ;
Courant := âge père + 10 ;

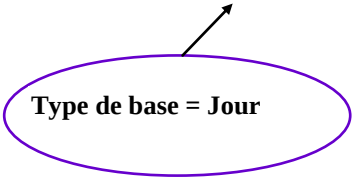
Une seule restriction est imposée aux variables du type âge : elles ne pourront se voir affecter des valeurs sortant de l'intervalle prévu.

Ainsi l'affectation courant := âge père + 10 entraînera une erreur si la valeur âge père + 10 est supérieure à 120. Même résultat pour l'instruction READ précédente, si l'utilisation fournit une valeur en dehors des limites 0 et 150

L'exemple précédant défini un type intervalle à partir d'un type entier (type de base ou hôte), mais le type de base peut être n'importe quel type ordinaire.

Exemple :

TYPE jour = (Samedi, Dimanche, Lundi, Mardi, Mercredi, Jeudi, Vendredi) ;
Jour trav = Samedi ... Mercredi ;
Weekend = Jeudi ... Vendredi ;



Là encore, les variables déclarées de type jourtrav ou weekend pourront être manipulées comme n'importe quelle valeur du type jour.

Ainsi, on peut déclarer comme suite :

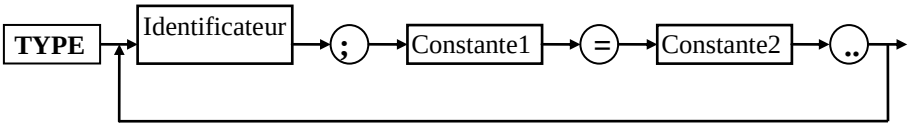
VAR Aujour : jourtrav ;
courant : jour ;

Les affectations suivantes sont correctes :

Au jour := Mercredi ;
Courant := SUCC (au jour) ;

- Il est interdit d'écrire TYPE Semaine = Jeudi... Samedi ; car l'ordre d'énumération dans Jour est différent.

2- Syntaxe et règle d'écriture :



Avec : Constante1 et Constante2 deux constantes d'un même type ordinal tel que Constante1 < Constante2.

- Notez que le type de base peut être n'importe quel type ordinal : prédéfini ou énuméré.

PROPRIÉTÉS DU TYPE INTERVALLE :

Les variables d'un type intervalle jouissent des mêmes propriétés que celles du type hôte ou de base, elles peuvent intervenir dans les mêmes expressions.

Les variables d'un type intervalle ne diffèrent des variables du type hôte que sur un point : Elles ne peuvent se voir affecter des valeurs situées en dehors de l'intervalle imposé.

Exemple:

```
PROGRAM  DEBOR_IN ;

TYPE  âge = 1 .. 120 ;

VAR    âge_fils, âge_père : âge ;

BEGIN

âge_fils := 80 ;
âge_père := âge_fils + 70 ;
```

Vous voyez que l'âge du père n'a pas dépassé la borne supérieure de l'intervalle 1.. 120.

Exemple :

```
PROGRAM EXEMPLE;
Type Jour = (Samedi,Dimanche,Lundi,Mardi,
Mercredi,Jeudi,Vendredi) ;
      Travail=Smedi..Mercredi ;
      Week_end=Jeudi..Vendredi ;
VAR         J :Jour ;
            n :Integer ;
BEGIN
n :=1
FOR J: =Samedi to Mercredi
DO BEGIN
       WRITELN (″On travail″,n,″fois″);
       n := n+1 ;
       END ;
n := 1
FOR    J := Jeudi To Vendredi
      DO BEGIN
       WRITELN(″On se repose″,n,″fois″);
      n :=n +1 ;
      END ;
END.
```

```
On travail 1fois
On travail 2fois
On travail 3fois
On travail 4 fois
On travail 5 fois
On serepose1fois
Onserepose2fois
```

RÉSUMÉ :

En plus des types prédéfinis, le langage Pascal permet à l'utilisateur de définir ses propres types de données. La déclaration de un ou plusieurs types de données, se fait dans la partie du programme réservée à la déclaration de tous les types envisagés par l'utilisateur et doit se précéder par le mot réservé **TYPE**.

Cette leçon a fait l'objet de l'étude de deux types. Le premier appelé type énuméré, il donne à l'utilisateur la possibilité de définir un ensemble ordonné de valeurs que pourra prendre une variable. Le second appelé type intervalle, définit un ensemble de valeurs comprises entre deux limites.

Ainsi, toute variable de ces deux types doit prendre des valeurs parmi ceux du type énuméré ou comprises entre les bornes de l'intervalle. Toute affectation d'une autre valeur au delà est incorrecte.

IV- EXERCICES D'APPLICATION :

EXERCICE N°01 :

Déterminer les éventuelles erreurs commises dans les déclarations suivantes :

a) TYPE positif = 0... MAXINT ;

b) TYPE préfixe = (sur, sous, super, in, im) ;

c) TYPE bonombre = (3, 5, 7, 11, 13) ;

d) TYPE lettre = '1'... 'n' ;

e) TYPE déb_alph = 'a'... 'p' ;

fin_alph = 'm' ... 'z' ;

f) TYPE voyelle = (a, e, i, o, u, y) ;

Remarque : Maxint est de type entier et vaut 32767.

EXERCICE N°02 :

Soit une déclaration de type énuméré : TYPE Feu = (vert, orange, rouge) ;

Ecrire un programme qui permet d'entrer une valeur de 0 à 2 pour déterminer l'ordre du feu sélectionné.

EXERCICE N°03 :

Soit les déclarations suivantes **:**

TYPE Semaine = (Samedi, dimanche, Lundi, Mardi, Mercredi);

Week = (Jeudi, Vendredi) ;

Ecrire le programme qui permet de déterminer l'ordre d'un jour de semaine et l'ordre d'un jour de week end.

EXERCICE N°04 :

Ecrire un programme qui permet de lire un numérique entre 0 et 9 et un caractère entre ‘A’ et ‘Z’ et les afficher.

V- CORRECTION DES EXERCICES :

EXERCICE N°01 :

b) **in** est un mot clé

c) 3, 5, 7, 11, 13 ne sont pas des identificateurs corrects, se sont des constantes entières.

Remarque : f) est correcte, bien que a, e, i, o, u et y sont de simples identificateurs n’ayant en fait rien à voir avec les caractères ‘a’, ‘e’, ‘i’, ‘o’, ‘u’ et ‘y’.

EXERCICE N°02 :

```
PROGRAM EXO2 ;
TYPE        Feu = (vert, orange, rouge) ;
VAR                 Val: INTEGER;

BEGIN

CLRSCR ;   (* Efface l’écran *)
WRITE (‘Entrez une valeur entre 0 et 2 :’) ;
READLN (Val);

IF val = ORD (orange)
    THEN  WRITELN (‘! PRUDENCE’, ORD (orange));

IF val = ORD (vert)
    THEN  WRITELN (‘⟶   PASSEZ’, ORD (vert));

IF val = ORD (rouge)
    THEN  WRITELN (‘⟶  STOP ⟵  ’, ORD (rouge) ) ;

END.
```

EXERCICE N°03 :

```
PROGRAM EXO3 ;

TYPE    Semaine = (Samedi, Dimanche, Lundi, Mardi, Mercredi) ;
        Week = (Jeudi, Vendredi) ;
VAR     Jour : Semaine ;
        Num: Week;
        J: INTEGER;
BEGIN

CLRSCR

Num := Jeudi ;

WRITELN (‘Ordre de week end:’ ORD (Num));
Num := SUCC(Num) ;

WRITELN (‘Ordre de week end:’ ORD (Num));

Jour: = Samedi;

FOR J: = 0 TO 4

    DO  BEGIN
        WRITELN (‘Ordre de jour :’, ORD(Jour)) ;
         IF J < 4
            THEN    Jour: = SUCC (Jour);
         END;
END.
```

REMARQUE :

1. Ici, nous avons testé la valeur de J pour ensuite appeler la fonction SUCC car le successeur de **Mercredi** n'est pas défini.

2. Il n'est pas possible d'effectuer une lecture ou une écriture dans une variable de type scalaire par énumération, comme par exemple READ(Num).

EXERCICE N°04:

```
PROGRAM EXO4;

TYPE      Numeric = 0... 9;
          Alphabet = 'A'... 'Z';

VAR       N: Numeric;
          A : Alphabet ;

BEGIN

WRITELN (' Donnez un numérique ') ;
READ (N) ;
WRITELN (' Donnez une lettre alphabétique ') ;
READ (A);
WRITELN (N, A);

END.
```